APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTORS:

Jinquan DAI
Shanghai, China

Luddy HARRISON
Chestnut, Massachusetts

Cotton SEED
Cambridge, Massachusetts

Bo HUANG
Shanghai, China

TITLE: LIVE SET TRANSMISSION IN PIPELINING APPLICATIONS

ASSIGNEE: Intel Corporation
Santa Clara, California

ATTORNEYS/ AGENTS:

Venable LLP
P.O. Box 43485
Washington, DC 20043-9899
Telephone: (202) 344-4000
Facsimile: (202) 344-8300

ATTORNEY DOCKET NO.: 42339-200559

BACKGROUND OF THE INVENTION

Some processors, for example, some network processors, may include highly parallel architectures. Accordingly, such processors may be well-suited for use with applications that take advantage of these parallel architectures. For example, a network application for packet processing may include, for example, receiving a packet, routing table look-up, and enqueueing of the packet. Such an application may be parallelized through pipelining and multi-threading transformations.

In implementing such a transformation, the data may typically be transmitted from stage to stage via a global resource ("pipe", which may refer to one or more various data transmission channels). However, not all data transmitted from stage to stage may be necessary. As a result, unnecessary overhead may be incurred in transmitting data that are not needed.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention will now be described in connection with the associated drawings, in which:

Figures 1A and 1B show a conceptual example of how an application may be handled according to embodiments of the invention;

Figures 2A and 2B show an example of an application and how it may be transformed according to embodiments of the invention;

Figures 3A, 3B, and 3C use the example of Figures 2A and 2B to show different ways of transmitting data between stages, according to various embodiments of the invention;

Figure 4 shows an example of traversal of a concatenated control flow graph according to some embodiments of the invention;

Figure 5 depicts a flowchart showing a process according to an embodiment of the invention;

Figure 6 shows a control flow graph that may be generated as an intermediate product of the flowchart of Figure 5, according to an embodiment of the invention;

Figures 7A and 7B show concatenated control flow graphs demonstrating interference according to some embodiments of the invention; and

Figure 8 depicts a conceptual block diagram of a system for implementing some embodiments of the invention.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the invention are discussed in detail below. While specific exemplary embodiments are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations can be used without parting from the spirit and scope of the invention.

Components/terminology used herein for one or more embodiments of the invention are described below:

In some embodiments, "computer" may refer to any apparatus that is capable of accepting a structured input, processing the structured input according to prescribed rules, and producing results of the processing as output. Examples of a computer may include: a computer; a general purpose computer; a supercomputer; a mainframe; a super mini-computer; a mini-computer; a workstation; a microcomputer; a server; an interactive television; a hybrid combination of a computer and an interactive television; and application-specific hardware to emulate a computer and/or software. A computer may have a single processor or multiple processors, which may operate in parallel and/or not in parallel. A computer may also refer to two or more computers connected together via a network for transmitting or receiving information between the computers. An example of such a computer may include a distributed computer system for processing information via computers linked by a network.

In some embodiments, a "machine-accessible medium" may refer to any storage device used for storing data accessible by a computer. Examples of a machine-accessible medium may include: a magnetic hard disk; a floppy disk; an optical disk, such as a CD-ROM or a DVD; a magnetic tape; a memory chip; and a carrier wave used to carry machine-accessible electronic data, such as those used in transmitting and receiving e-mail or in accessing a network.

In some embodiments, "software" may refer to prescribed rules to operate a computer. Examples of software may include: code segments; instructions; computer programs; and programmed logic.

In some embodiments, a "computer system" may refer to a system having a computer, where the computer may comprise a machine-accessible medium embodying software to operate the computer.

Figure 1A shows an example of an application 10 that may be pipelined and parallelized as in embodiments of the present invention. This particular example depicts an exemplary ingress operation for a networking protocol. In this application, three operations may be executed sequentially for each data packet. However, the nature of these operations may be such that several packets may be processed simultaneously, if the hardware permits. For example, if the computing platform includes multiple processors, each processor, in turn, comprising multiple sub-processors that may provide parallel operations, this computing platform may be suited to such operations; however, other computing platforms may also be suitable, and the invention is not to be thusly limited. In such a case, the application may be pipelined, for example, in three stages, 10a, 10b, and 10c, as shown in Figure 1B, in which each stage may provide for execution of one of the three operations on multiple packets. In carrying this out, necessary data may be passed from each stage to the next succeeding stage.

Figures 2A and 2B depict a more general example of the pipelining and parallelizing of a program, according to embodiments of the invention. Figure 2A shows a control flow graph (CFG) of an application program 20. In block 21, a variable t1 may be computed, followed by a condition based on t1. Depending on the outcome of the condition, the application may continue with either block 22 or block 23, in each of which a function of the variable t1 may be computed. In block 24, another variable, t2, may be defined, and a further condition (also based on t1) may be executed. Depending upon the outcome of the condition in block 24, the application may continue with either block 25, where another variable, t3, may be defined and a function of t3 may be computed, or with block 26, where a function of t2 may be computed.

In Figure 2B, the application 20 of Figure 2A has been partitioned into two stages, 20a and 20b, to be referred to as stage 1 and stage 2. In particular, in stage 1 (20a), block 21a may contain the same operations as for block 21 of application 20. If the condition of block 21a comes out such that the left branch, block 22a, is chosen, block 22a may include the same operation as in block 22 of application 20. Otherwise, block 23a may contain no operation. Block 24a may be identical to block 24, and the condition may determine whether stage 1 continues with block 25a or block 26a. If it continues with block 25a, the variable t3 may be defined, as in block 25. If block 26a is chosen, then no operation may be executed. From stage 1 (20a), the three data defined, variables t1, t2, and t3, may be passed to stage 2 (20b) via a pipe 27.

In stage 2 (20b) of Figure 2B, block 21b may execute only the condition of block 21; there may be no need to re-compute t1 because it may have been obtained from stage 1. If the condition determines that stage 2 should continue with block 22b, no operation may be executed. This is consistent with the partitioning, as the function defined in block 22 may be carried out in block 22a of stage 1. However, if the condition determines that stage 2 should continue with block 23b, the same operation as shown in block 23 of application 20 may be executed. Then, stage 2 may continue to block 24b, where only the condition may be executed, to determine if stage 2 may continue with block 25b or with block 26b. Again, because t2 may have already been defined, there may be no need to re-compute it. If stage 2 continues with block 25b, the function of t3 shown in block 25 of application 20 may be executed (again, t3 may have been previously defined). On the other hand, if stage 2 continues with block 26b, the function of t2 shown in block 26 of application 20 may be executed.

The transmission of only a "live set" of data, that is, a set of data that are "alive" at a boundary between stages of a partition of the application (created by pipelining) may serve to improve efficiency of execution of the pipelined and parallelized application. Data that are "alive" are those data already determined and to be used in a subsequent stage (for example, if a variable is defined in a first stage, not used in a second stage, used in a third stage, and not used subsequently, it would be alive at the boundaries between the first and second stages and the second and third stages, but not at the boundary between the third and fourth stages). In the example of Figure 2B, all three data, t1, t2, and t3, are alive at the boundary between stage 1 (20a) and stage 2 (20b).

Figures 3A-3C show three ways, according to various embodiments of the invention, in which the live set of data may be transmitted between two stages of a pipeline. Figures 3A-3C use the example shown in Figures 2A and 2B as an illustration of the various embodiments of the invention.

Figure 3A shows an example of what may be termed "conditional live set transmission." In this embodiment, each datum of the live set may be conditionally transmitted. That is, using the example of Figure 3A, if the datum is needed in a block of stage 2 (30b), that datum may be placed into the pipe at a point following its definition in stage 1 (30a) and may be obtained from the pipe at a corresponding point in stage 2 (30b). Only data that are needed in any subsequent stage may be transmitted. In particular, in stage 1 (30a), in blocks 31a, 32a, and 33a, respectively, the "pipe_put" instructions may place the data, t1, t2, and t3, respectively, into the pipe, while in stage 2 (30b), in blocks 31b, 32b, and 33b, respectively, the "pipe_get" instructions may obtain the data from the pipe.

Figure 3B shows an example of what may be termed "naively unified live set transmission." In this embodiment of the invention, all data may be transmitted to the pipe at a single point of each stage and may be obtained from the pipe at a single point of each stage. This technique may be advantageous over the embodiment of Figure 3A in that the synchronization required for the type of data transmission shown in Figure 3B may be less than that required for the type of data transmission shown in Figure 3A, due to the fact that, contrasting the lines 37 in Figure 3A and 38 in Figure 3B, the critical section of the application centered around pipe operations may be significantly larger in the embodiment of Figure 3A than in the embodiment of Figure 3B. As a result, efficiency may be improved by using the embodiment of Figure 3B.

In Figure 3B, as each datum is determined, that datum may be duplicated in a new datum that may be used for transmission. In Figure 3B, this may be seen in the presence in stage 1 (30a') of the definitions of t4 = t1, t5 = t2, and t6 = t3 in blocks 31a', 32a', and 33a', respectively. Then, at the end of a stage, all of the data may be placed into the pipeline together; in Figure 3B, this may be seen in the pipe_put instruction in block 34a' of stage 1 (30a'). At the beginning of each subsequent stage, as demonstrated by block 31b' of stage 2 (30b'), all of the data may be retrieved from the pipe together (noting the pipe_get instruction). As reflected in blocks 32b', 33b', 35b', and 36b' of stage 2 (30b'), the instructions that may be executed in subsequent stages may be adapted to operate on the new data (t4, t5, and t6) instead of on the original data (t1, t2, and t3).

While, as discussed above, the embodiment of Figure 3B may provide an improvement over the embodiment of Figure 3A, the embodiment of Figure 3C may, in turn, provide improvement over the embodiment of Figure 3B. The naïve unified live set transmission of Figure 3B may transmit more data than necessary because, due to the parallelization, objects that appear to both be part of the live set may, in fact, not both be alive simultaneously. For example, returning to Figures 2A and 2B, one may notice that, as a result of the condition executed in block 24, either a function of t3 may need to be computed (block 25) or a function of t2 may need to be computed (block 26). Noting that t3 may be computed in stage 1 (20a), if needed, it may not be necessary to pass both t2 and t3 to stage 2 (20b); that is, t2 and t3 may not both be alive at the boundary between stage 1 (20a) and stage 2 (20b), even though, at first glance, they may both appear to be alive. In the embodiment of Figure 3C, such redundancy may be eliminated, thus taking advantage of the parallelism.

The embodiment of Figure 3C may take advantage of the above-described situation by defining new data, t4 and t5, in blocks 31a', 32a', and 33a" of stage 1 (30a"), which may be placed in the pipe at block 34a". The values of t4 and t5 may be obtained from the pipe in block 31b" of stage 2 (30b"). Similarly to the embodiment of Figure 3B, blocks 31b", 32b', 33b", 35b', and 36b' of stage 2 (30b") may be adapted to operate on the new data, in this case, t4 and t5. The difference between this embodiment and the embodiment of Figure 3B is that, instead of defining t4 = t1, t5 = t2, and t6 = t3, this embodiment may define t4 = t1 and t5 = either t2 or t3. That is, the embodiment of Figure 3C may take advantage of the condition of blocks 32a' and 32b' to determine whether the value of t2 or the value of t3 may be needed in stage 2 (30b"). If the condition comes out one way, t3 may be computed in block 33a" and t5 may be set equal to t3; then, in stage 2 (30b"), in block 33b", h(t5) may be computed, and there may be no need for t2. On the other hand, if the condition comes out the other way, t5, which was set equal to t2 in block 32a', is used in block 36b' to compute k(t5), and there may be no need for t3. As a result, there may be less data to transmit from stage to stage.

The criterion for determining whether or not two data may be used in subsequent stages of a pipelined, parallelized application, as discussed above, may be whether the two data are simultaneously alive. In such a case, the data are said to "interfere" with each other. That is, if two data interfere with each other, they may need to both be passed on through the data pipe. Figure 4 shows a concatenated CFG of the pipelined, parallelized stages of Figure 2B. One way of determining whether or not there may be interference between data may be to traverse all possible paths of the concatenated CFG of Figure 4. However, a naïve traversal of the concatenated CFG of Figure 4 may result in paths that may never occur, such as that shown by the heavy arrow in Figure 4. The result of this may be that data that in reality may not interfere may be found to interfere.

To solve this problem, embodiments of the invention may use a procedure illustrated by the flowchart of Figure 5. Blocks 51 and 52 form what may be considered to be a first component of the procedure of Figure 5. In block 51, for each live object, v, a new object v' may be defined immediately following the point in the CFG at which v is first defined. Then, in block 52, each of the newly-defined object (e.g., v') may substituted for the occurrences of the original objects (e.g., v), in subsequent stages. Using the example of Figure 2A, the modified CFG of Figure 6 may be obtained (prior to separation into stages) when blocks 51 and 52 of Figure 5 are applied.

Continuing with the procedure of Figure 5, blocks 53 and 54 together may be considered to form a second component of the procedure, which may be used to compute a correct interference relationship between live objects over the concatenated CFG of the stages, without impossible paths (as discussed in connection with Figure 4, above). In block 53, an interference relationship may be determined between the various newly-defined objects (e.g., u' and v') for the live objects (e.g., u and v). If it is determined that two newly-defined objects interfere, then their corresponding live objects may also be determined to interfere in a final interference graph (to be discussed further below). In block 54, if there is a path from a use of one newly-defined object (e.g., v') to a definition of another newly-defined object (e.g., u'), then the corresponding live objects (e.g., u and v) may be determined to interfere in the final interference graph.

The procedure of blocks 53 and 54 may be illustrated via the concatenated CFGs shown in Figures 7A and 7B. Figures 7A and 7B correspond to the example of Figure 2B (and Figure 4). In Figures 7A and 7B, the points U1, V1, U2, V2, W1, and W2 may be defined as follows:

Name	Definition
V1	Definition point of a first variable (e.g., v) in the first stage.

U1	Definition point of a second variable (e.g., u) in the first stage.
V2	Point in the second stage that corresponds to V1 point of the first stage.
U2	Point in the second stage that corresponds to U1 point of the first stage.
W2	A use of the first variable (e.g., v) in the second stage.
W1	Point in first stage that corresponds to W2 point of the second stage.

These definitions may provide a basis by which to analyze the interference of two quantities (e.g., v and u). Note that W2 (and, accordingly, W1) may be moved from one use of the first variable (e.g., v) in the second stage to another as the procedure continues with determining whether the two quantities interfere (that is, all uses may be considered). Also note that, while these definitions are given for the example of a two-stage partition (i.e., as in the example of Figures 7A and 7B), they may be generalized for partitions of more than two stages. For the purpose of determining whether or not there is interference between two quantities, v and u, it may be shown that v and u will interfere if and only if the order in which the above-defined points occur satisfies one of two conditions: (1) V1 -> U1 -> W1 -> V2 -> U2 -> W2; or (2) V1 -> W1 -> U1 -> V2 -> W2 -> U2.

Suppose, in the example of Figure 2B (and Figure 4), that the quantities t1 and t2 are considered to determine if they interfere with each other. This is the example illustrated in Figure 7A. In this case, one may let V1 correspond to the point at which t1 is defined in the first stage and U1 correspond to the point at which t2 is defined in the first stage. Moving to the second stage, then, the corresponding points may be marked by V2 and U2, respectively. Then, W2 may, for example, be placed as shown, and the corresponding point in the first stage may be marked by W1. In this case, the six points follow the order V1 -> U1 -> W1 -> V2 -> U2 -> W2, and thus, t1 and t2 interfere.

Now, in the same example, consider the quantities t1 and t3, which is illustrated in Figure 7B. Again, V1 may correspond to the point at which t1 is defined in the first stage, and U1 may correspond to the point at which t3 is defined in the first stage. The remaining points (V2, U2, W2, and W1) may be placed in the same manner as before. The resulting order of these points is V1 -> W1 -> U1 -> V2 -> W2 -> U2, and as a result, t1 and t3 interfere.

Returning to Figure 5, blocks 55-57 may be considered as forming a third component of the procedure. As a result of the determinations made in blocks 53 and 54, an interference graph may be formed. In this interference graph, each node of the graph may represent one of the variables (e.g., t1, t2, and t3), and edges may be provided between nodes if the corresponding variables have been determined to interfere with each other. In block 55, the interference graph may be colored, using any known or as yet to be discovered graph coloring algorithm.

In block 56, the colored interference graph may be considered. For each live object v, if it is colored using a color k, a new instruction defining v_k may be inserted immediately following the definition point of v in the CFG of the first stage (or the first stage in which v is defined). Then, at the end of the stage, all the v_k defined for that stage may be transmitted to the pipe.

Block 57 may then be used to examine one or more subsequent stages. For each live object v colored with a color k, in the subsequent stage, each use of v may be changed to a use of v_k in that (subsequent) stage. All of the v_k may be received from the pipe at the beginning of each (subsequent) stage.

Some embodiments of the invention, as discussed above, may be embodied in the form of software instructions on a machine-accessible medium. Such an embodiment is illustrated in Figure 8. The computer system of Figure 8 may include at least one processor 82, with associated system memory 81, which may store, for example, operating system software and the like. The system may further include additional memory 83, which may, for example, include software instructions to perform various applications. System memory 81 and additional memory 83 may comprise separate memory devices, a single shared memory device, or a combination of separate and shared memory devices. The system may also include one or more input/output (I/O) devices 84, for example (but not limited to), keyboard, mouse, trackball, printer, display, network connection, etc. The present invention may be embodied as software instructions that may be stored in system memory 81 or in additional memory 83. Such software instructions may also be stored in removable or remote media (for example, but not limited to, compact disks, floppy disks, etc.), which may be read through an I/O device 84 (for example, but not limited to, a floppy disk drive). Furthermore, the software instructions may also be transmitted to the computer system via an I/O device 84, for example, a network connection; in such a case, a signal containing the software instructions may be considered to be a machine-accessible medium.

The invention has been described in detail with respect to various embodiments, and it will now be apparent from the foregoing to those skilled in the art that changes and modifications may be made without departing from the invention in its broader aspects. The invention, therefore, as defined in the appended claims, is intended to cover all such changes and modifications as fall within the true spirit of the invention.